Tema 12.3 - Creación de un Periférico PLB

Sistemas Electrónicos para Automatización

Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Hipólito Guzmán Miranda

Contenido

- Motivación
- Creación de un periférico PLB
- Otras funcionalidades
- Añadir periférico al diseño
- Modificaciones hardware
- Modificaciones software

Contenido

- Motivación
- Creación de un periférico PLB
- Otras funcionalidades
- Añadir periférico al diseño
- Modificaciones hardware
- Modificaciones software

Motivación

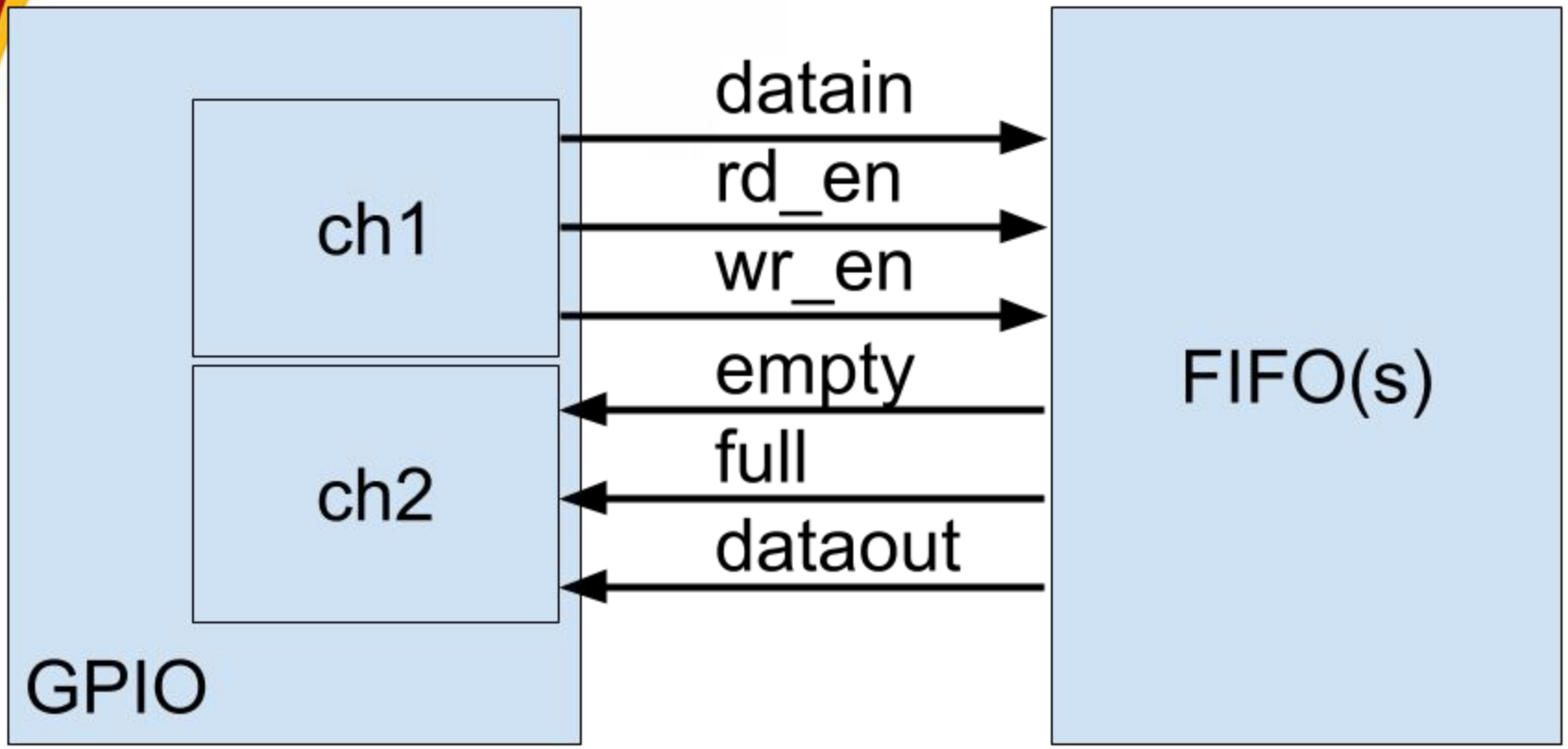
GPIO = quick and dirty solution :)

- GPIO es fácil y rápido de prototipar
  - ‘Sólo’ tienes que saber de FPGAs, microprocesadores empotrados, arquitectur microprocesadores, manejo de EDK, C, libr para manejo del GPIO… ;)
- Pero los accesos son lentos
- Y el interfaz es muy sencillo: si queremos tener interfaz tipo FIFO, regs, memoria
  - Tendríamos que implementarlos a mano: el será aún más lento

Ejemplo: acceso a una FIFO desde GPIO

- GPIO con dos canales
- Conectamos al canal de salida del GPIO las señales datain, wren, rden de la FIFO
- Conectamos al canal de entrada del GPIO las señales dataout, full, empty de la FIFO

Ejemplo: acceso a FIFO desde GPIO



Ejemplo: acceso a una FIFO desde GPIO

Operación de lectura:

- Comprobar bit empty (lectura GPIO in)
- Si !empty:
  - Activar rd_en (escritura en GPIO out)
  - Desactivar rd_en (escritura en GPIO out)
- Leer datos (lectura GPIO in)

4 operaciones para leer un dato!

Ejemplo: acceso a una FIFO desde GPIO

Operación de escritura:

- Comprobar bit full (lectura GPIO in)
- Si !full:
  - Escribir dato (escritura en GPIO out)
  - Activar wr_en (escritura en GPIO out)
  - Desactivar wr_en (escritura en GPIO out)

4 operaciones para escribir un dato!

Ejemplo: acceso a una FIFO desde GPIO

Adicionalmente, no tenemos control sobre cuántos ciclos se mantienen activos rd_en y wr_en, con lo cual escribimos/leemos múltiples veces los datos en la FIFO

Limitaciones del GPIO

- GPIO se puede usar cuando el interfaz es sencillo y no existen restricciones de tiempo fuertes
- En otro caso, representará un cuello de botella en nuestro sistema
- No obstante, puede ser muy útil para prototipado rápido de soluciones

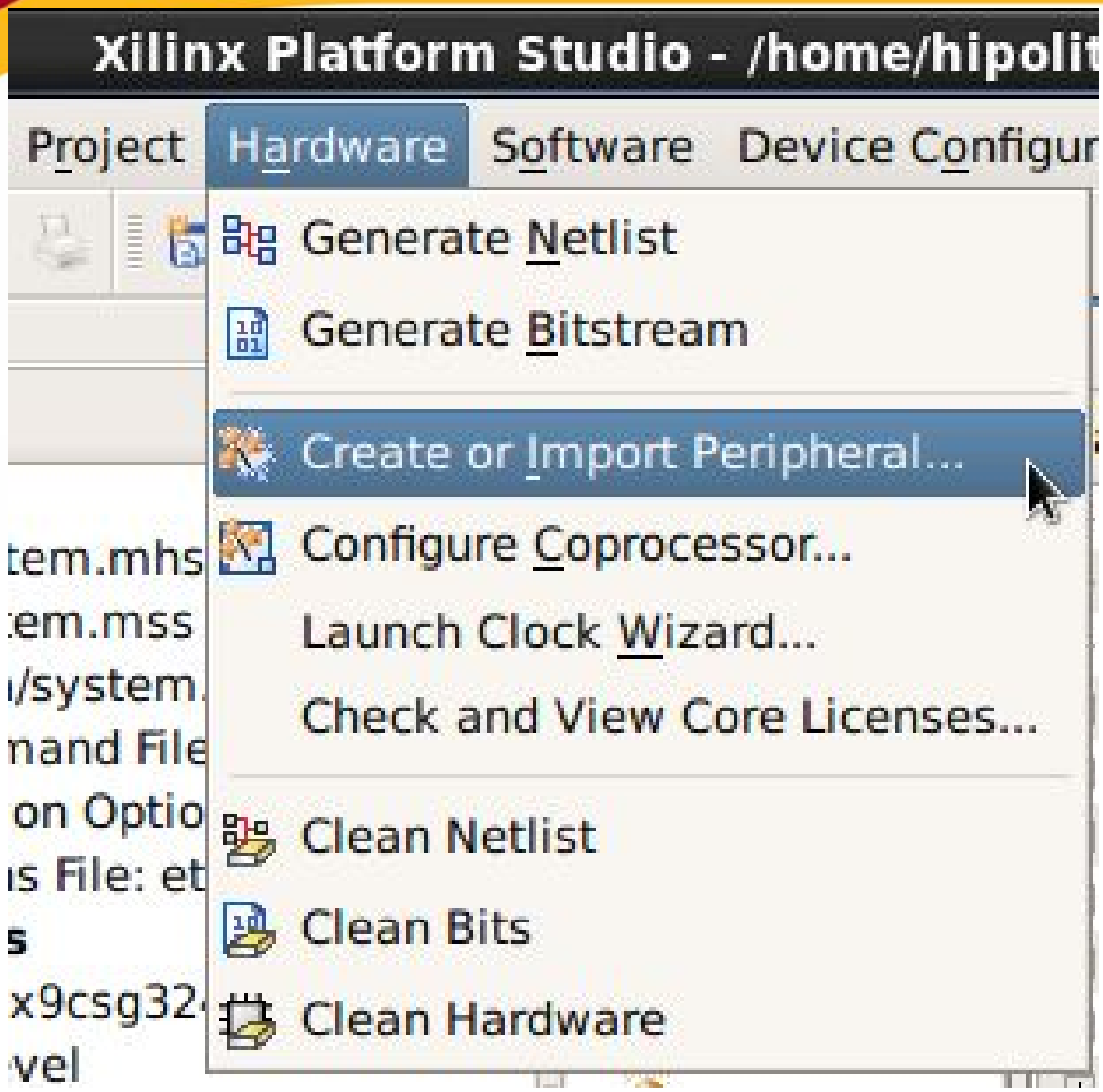
Contenido

- Motivación
- Creación de un periférico PLB
- Otras funcionalidades
- Añadir periférico al diseño
- Modificaciones hardware
- Modificaciones software

Periféricos PLB

- Mayores prestaciones de velocidad
- Interfaces más complejos:
  - Registros de usuario
  - FIFOS
  - Espacio RAM direccionable
- También: soporte para interrupciones

Creación de un periférico PLB



Create and Import Peripheral Wizard

Xilinx Embedded Processing Solutions

Xilinx Embedded Processing Solutions

XILINX

Welcome to the Create and Import Peripheral Wizard

This wizard will help you create and import a user EDK CoreConnect peripheral for use in processor systems developed using the EDK.

To continue, click Next.

More Info | < Back | Next > | Cancel

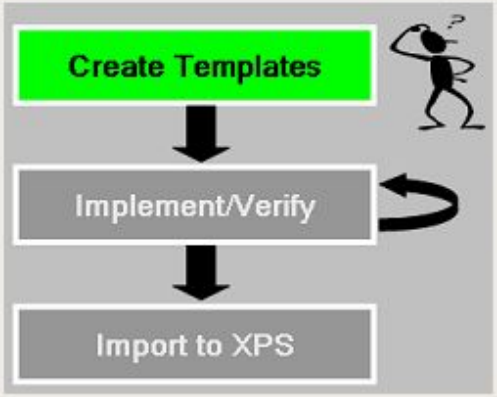
hipolito - File Browser | Xilinx Platform Studio ... | Create and Import Peri...

Create and Import Peripheral Wizard

Peripheral Flow

Indicate if you want to create a new peripheral or import an existing peripheral.

This tool will help you create templates for a new EDK CoreConnect peripheral, or help you import an existing EDK CoreConnect peripheral into an XPS project or EDK repository. The interface files and directory structures required by EDK will be generated.



Select flow

- (•) Create templates for a new peripheral
- ( ) Import existing peripheral

Flow description

This tool will create HDL templates that have the EDK compliant port/parameter interface. You will need to implement the body of the peripheral.

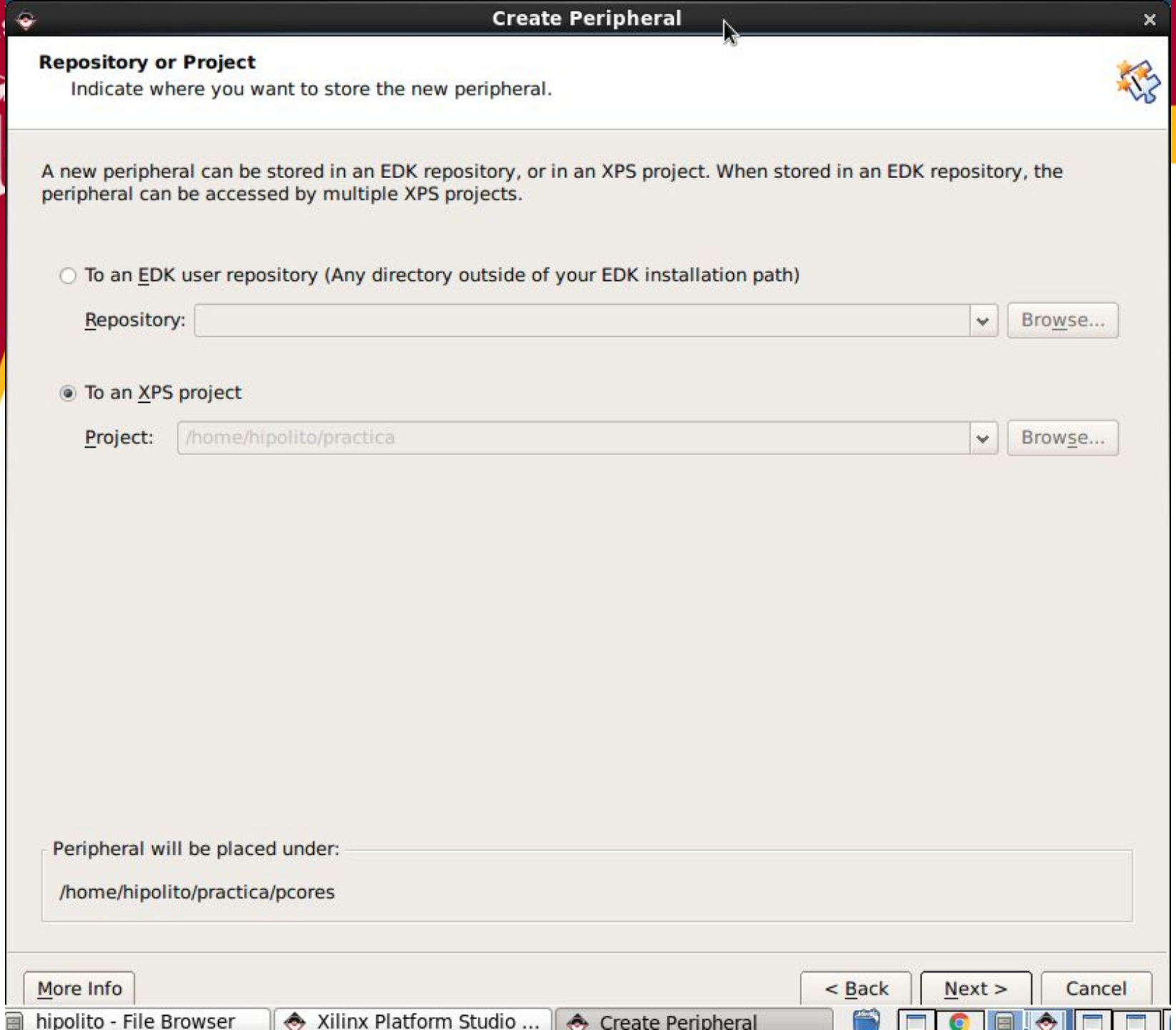
Options

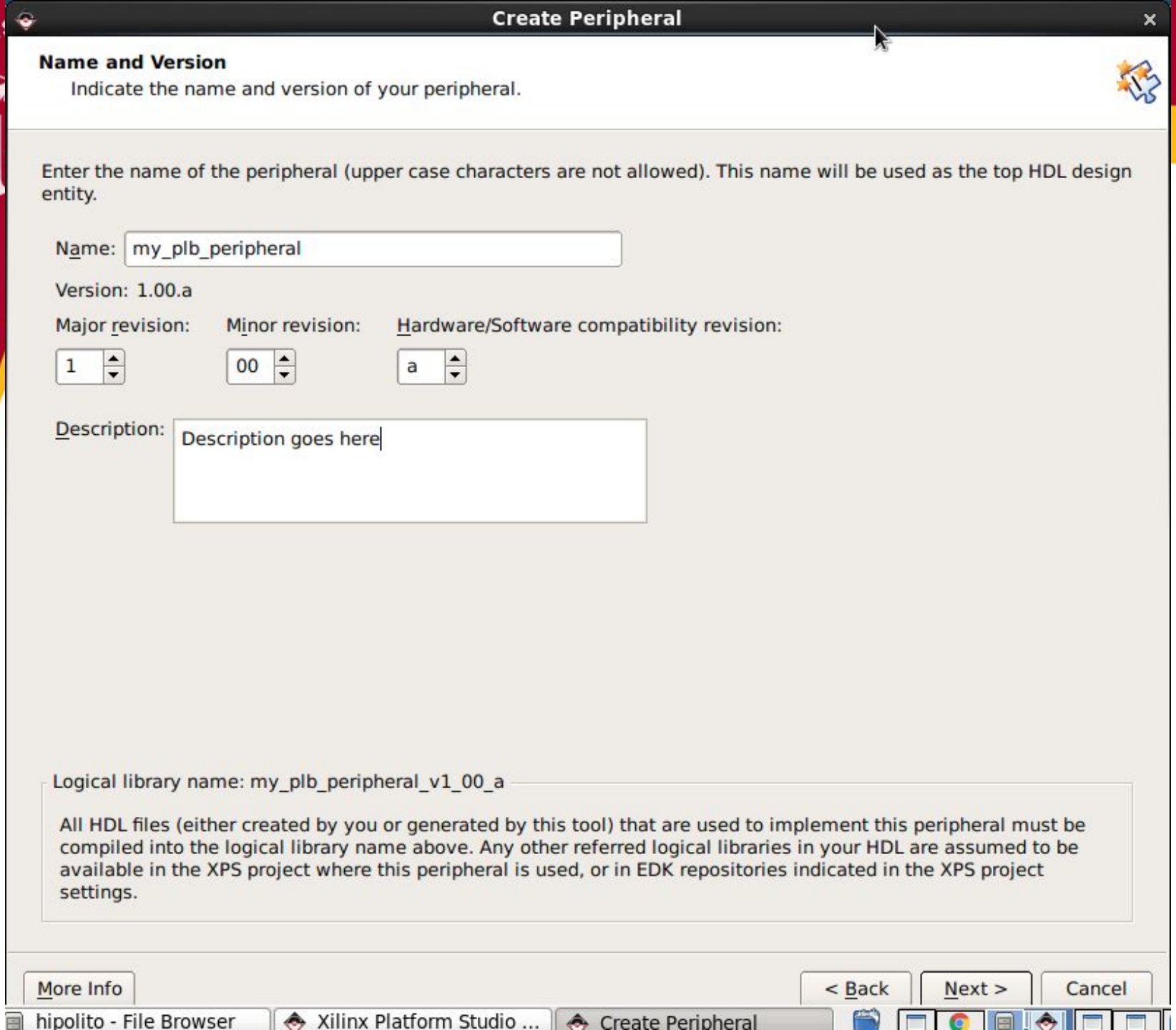
- [ ] Load an existing .cip settings file (saved from a previous session)

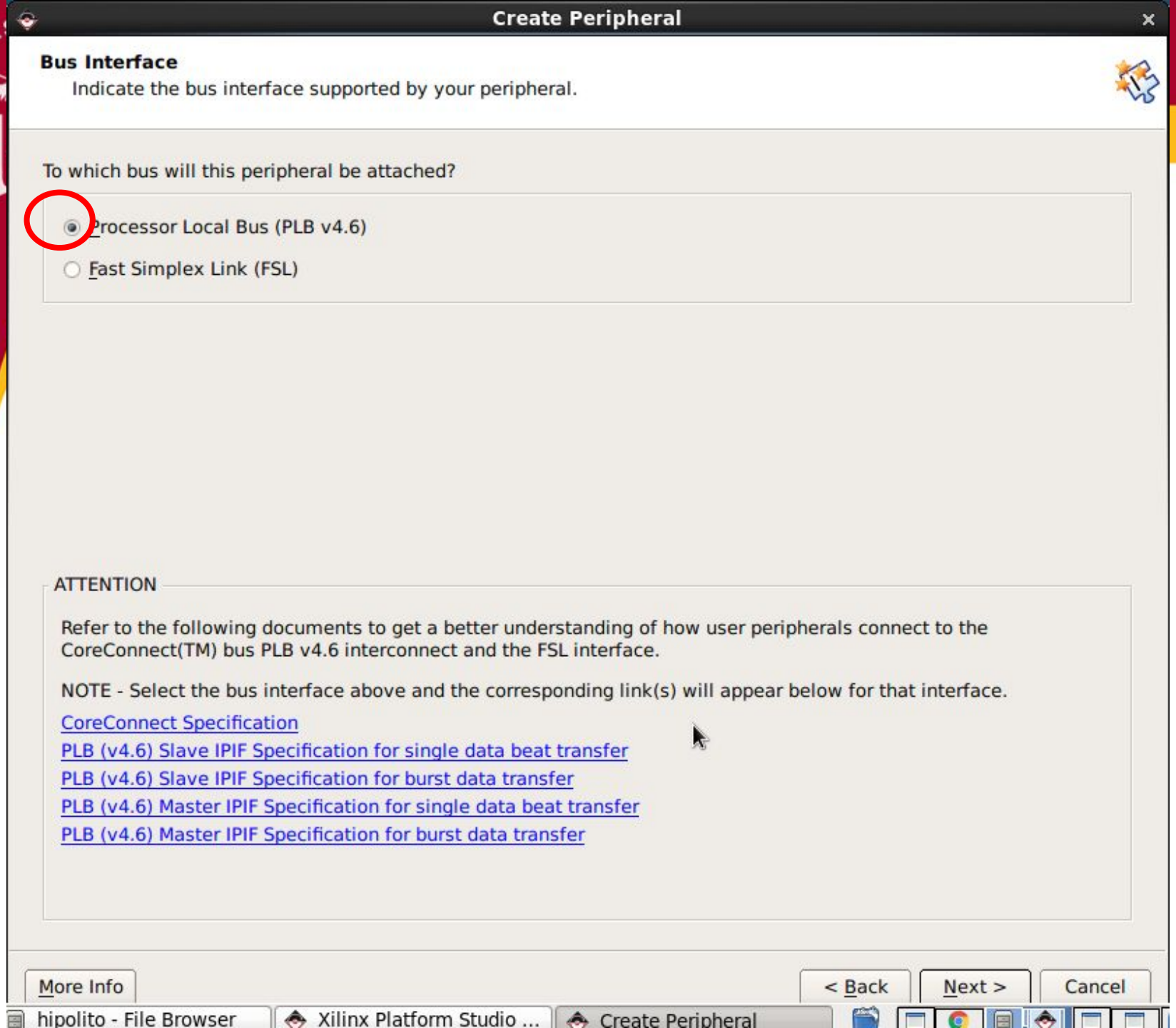
Browse...

More Info | < Back | Next > | Cancel

hipolito - File Browser | Xilinx Platform Studio ... | Create and Import Peri...


Create Peripheral
Repository or Project
Indicate where you want to store the new peripheral.
A new peripheral can be stored in an EDK repository, or in an XPS project. When stored in an EDK repository, the peripheral can be accessed by multiple XPS projects.
To an EDK user repository (Any directory outside of your EDK installation path)
Repository:
Browse...
To an XPS project
Project:
/home/hipolito/practica
Browse...
Peripheral will be placed under:
/home/hipolito/practica/pcores
More Info
< Back
Next >
Cancel
hipolito - File Browser
Xilinx Platform Studio ...
Create Peripheral


Create Peripheral
Name and Version
Indicate the name and version of your peripheral.
Enter the name of the peripheral (upper case characters are not allowed). This name will be used as the top HDL design entity.
Name: my_plb_peripheral
Version: 1.00.a
Major revision:
Minor revision:
Hardware/Software compatibility revision:
1
00
a
Description:
Description goes here
Logical library name: my_plb_peripheral_v1_00_a
All HDL files (either created by you or generated by this tool) that are used to implement this peripheral must be compiled into the logical library name above. Any other referred logical libraries in your HDL are assumed to be available in the XPS project where this peripheral is used, or in EDK repositories indicated in the XPS project settings.
More Info
< Back
Next >
Cancel
hipolito - File Browser
Xilinx Platform Studio ...
Create Peripheral

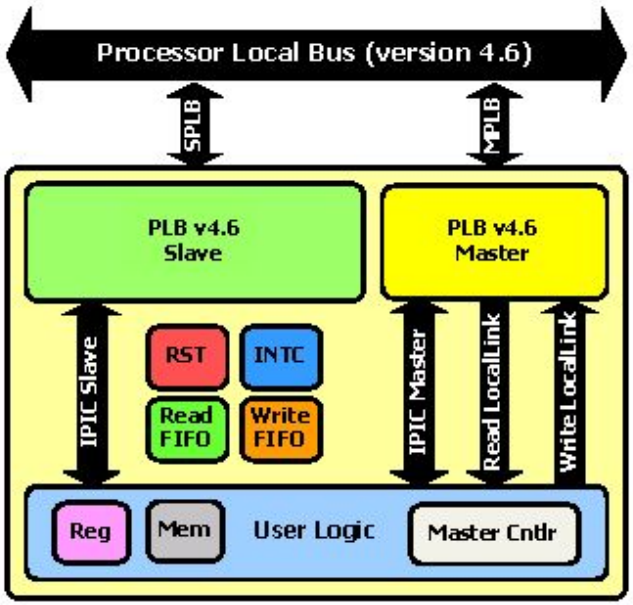

Create Peripheral
Bus Interface
Indicate the bus interface supported by your peripheral.
To which bus will this peripheral be attached?
Processor Local Bus (PLB v4.6)
Fast Simplex Link (FSL)
ATTENTION
Refer to the following documents to get a better understanding of how user peripherals connect to the CoreConnect(TM) bus PLB v4.6 interconnect and the FSL interface.
NOTE - Select the bus interface above and the corresponding link(s) will appear below for that interface.
CoreConnect Specification
PLB (v4.6) Slave IPIF Specification for single data beat transfer
PLB (v4.6) Slave IPIF Specification for burst data transfer
PLB (v4.6) Master IPIF Specification for single data beat transfer
PLB (v4.6) Master IPIF Specification for burst data transfer
More Info
< Back
Next >
Cancel
hipolito - File Browser
Xilinx Platform Studio ...
Create Peripheral

Create Peripheral

IPIF (IP Interface) Services

Indicate the IPIF services required by your peripheral.

Your peripheral will be connected to the PLB (v4.6) interconnect through corresponding PLB IP Interface (IPIF) modules, which provide you with a quick way to implement the interface between the PLB interconnect and the user logic. Besides the standard functions like address decoding provided by the slave IPIF module, the wizard tool also offers other commonly used services and configurations to simplify the implementation of the design.



Slave service and configuration

Typically required by most peripherals for operations like logic control, status report, data buffering, multiple memory/address space access, and etc. (PLB slave interface will always be included).

- [ ] Software reset
- [x] User logic software register
- [ ] Read/Write FIFO
- [ ] User logic memory space
- [ ] Interrupt control
- [ ] Include data phase timer

Master service and configuration

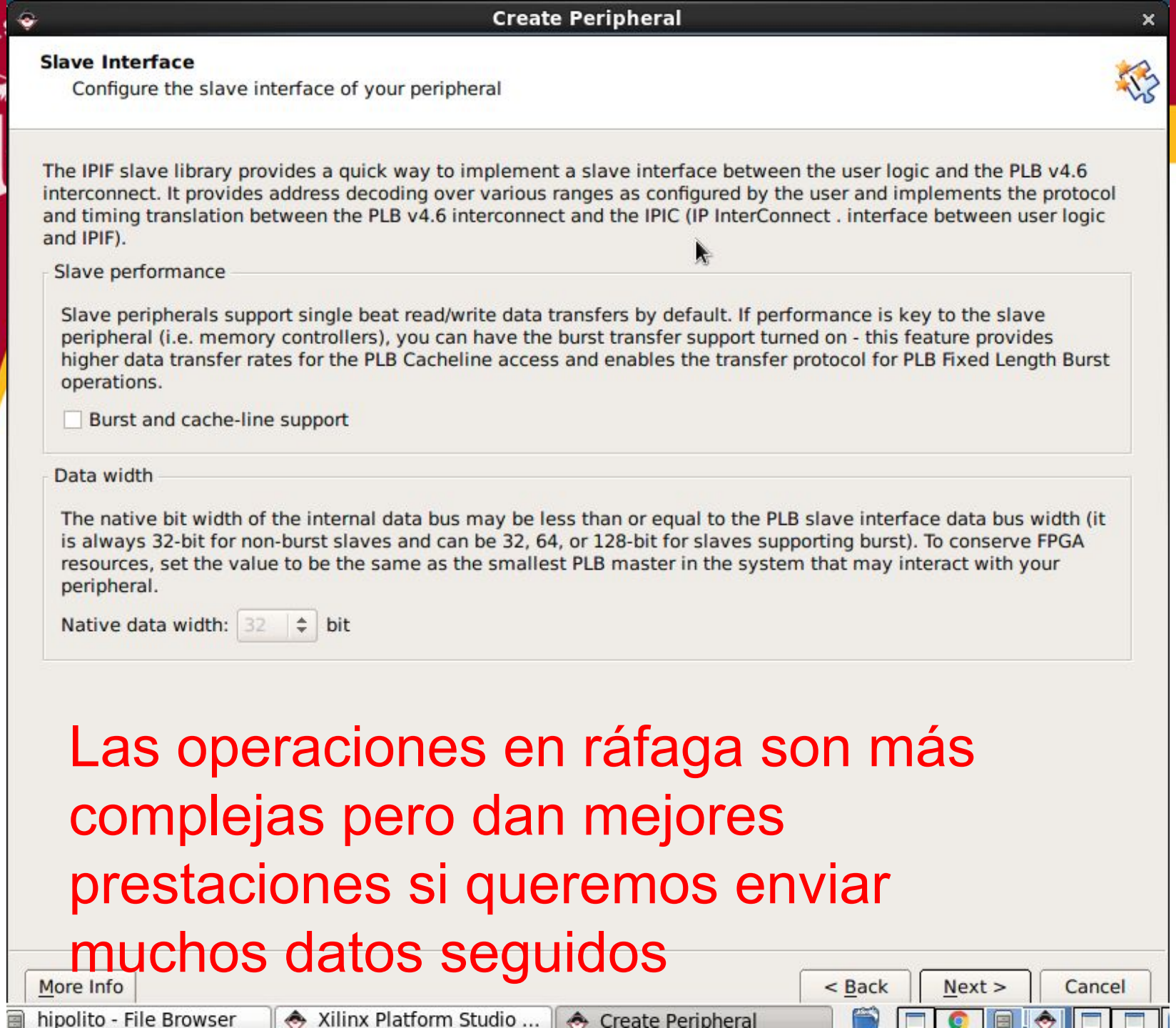
Typically required by complex peripherals like Ethernet and PCI for commanding data transfers between regions (PLB master interface will be included if master service selected).

- [ ] User logic master

Seleccionamos las funcionalidades del periférico

More Info | < Back | Next > | Cancel

hipolito - File Browser | Xilinx Platform Studio ... | Create Peripheral

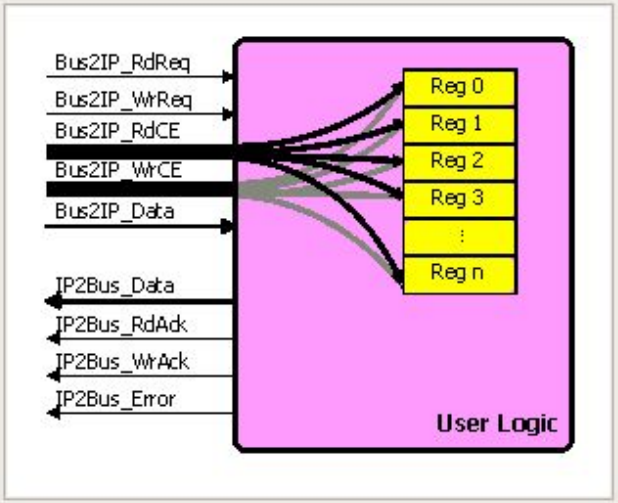

Create Peripheral
Slave Interface
Configure the slave interface of your peripheral
The IPIF slave library provides a quick way to implement a slave interface between the user logic and the PLB v4.6 interconnect. It provides address decoding over various ranges as configured by the user and implements the protocol and timing translation between the PLB v4.6 interconnect and the IPIC (IP InterConnect . interface between user logic and IPIF).
Slave performance
Slave peripherals support single beat read/write data transfers by default. If performance is key to the slave peripheral (i.e. memory controllers), you can have the burst transfer support turned on - this feature provides higher data transfer rates for the PLB Cacheline access and enables the transfer protocol for PLB Fixed Length Burst operations.
Burst and cache-line support
Data width
The native bit width of the internal data bus may be less than or equal to the PLB slave interface data bus width (it is always 32-bit for non-burst slaves and can be 32, 64, or 128-bit for slaves supporting burst). To conserve FPGA resources, set the value to be the same as the smallest PLB master in the system that may interact with your peripheral.
Native data width: 32 bit
Las operaciones en ráfaga son más complejas pero dan mejores prestaciones si queremos enviar muchos datos seguidos
More Info
< Back
Next >
Cancel
hipolito - File Browser
Xilinx Platform Studio ...
Create Peripheral

Create Peripheral

User S/W Register

Configure the software accessible registers in your peripheral.

The user specific software accessible registers will be implemented in the user-logic module of your peripheral. Such registers are typically provided for software programs to control and to monitor the status of your user logic. These registers are addressable on the byte, half-word, word, double word or quad word boundaries depending on your design. An example logic for register read/write will be included in the user-logic module generated by the wizard tool for your reference.

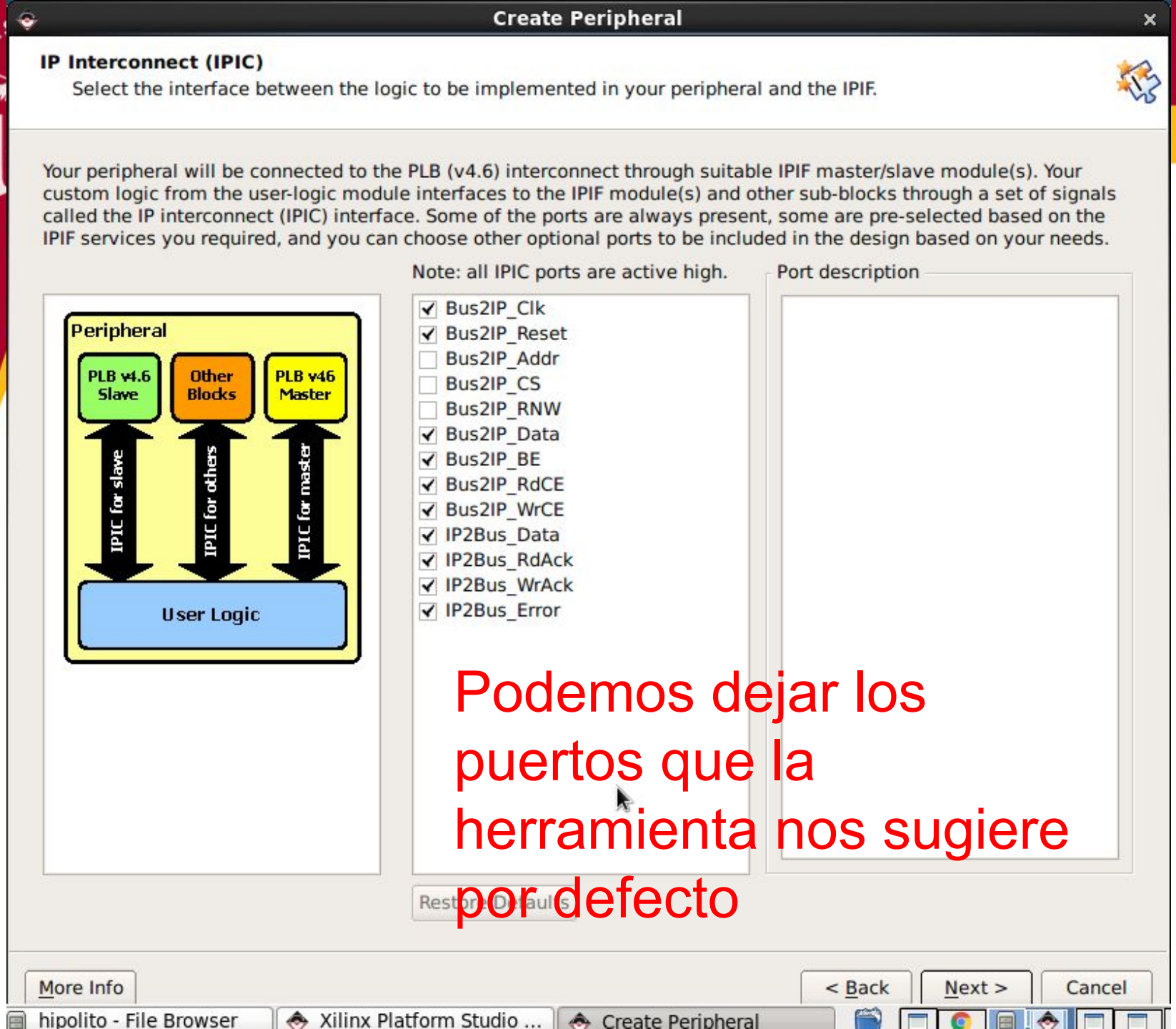


User logic software registers may take full advantage of the slave IPIF address-decoding service to generate CE decodes for all of the individual register of interest. The diagram on the left shows the simplest set of IPIC slave signals to read/write the registers.

Number of software accessible registers: 1 (1 to 4096)

More Info | < Back | Next > | Cancel

hipolito - File Browser | Xilinx Platform Studio ... | Create Peripheral

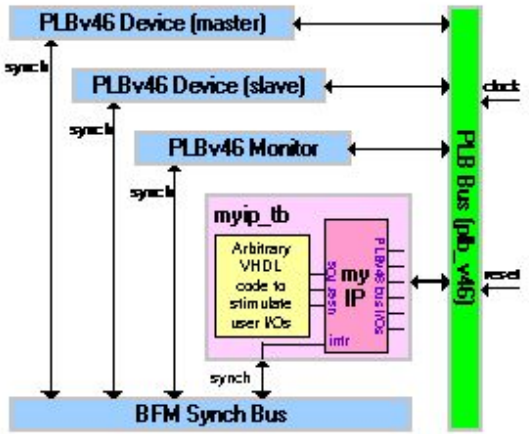

Create Peripheral
IP Interconnect (IPIC)
Select the interface between the logic to be implemented in your peripheral and the IPIF.
Your peripheral will be connected to the PLB (v4.6) interconnect through suitable IPIF master/slave module(s). Your custom logic from the user-logic module interfaces to the IPIF module(s) and other sub-blocks through a set of signals called the IP interconnect (IPIC) interface. Some of the ports are always present, some are pre-selected based on the IPIF services you required, and you can choose other optional ports to be included in the design based on your needs.
Peripheral
PLB v4.6 Slave
Other Blocks
PLB v46 Master
IPIC for slave
IPIC for others
IPIC for master
User Logic
Note: all IPIC ports are active high.
Bus2IP_Clk
Bus2IP_Reset
Bus2IP_Addr
Bus2IP_CS
Bus2IP_RNW
Bus2IP_Data
Bus2IP_BE
Bus2IP_RdCE
Bus2IP_WrCE
IP2Bus_Data
IP2Bus_RdAck
IP2Bus_WrAck
IP2Bus_Error
Port description
Restore Defaults
Podemos dejar los puertos que la herramienta nos sugiere por defecto
More Info
< Back
Next >
Cancel
hipolito - File Browser
Xilinx Platform Studio ...
Create Peripheral

Create Peripheral

(OPTIONAL) Peripheral Simulation Support

Generate optional files for simulation using Bus Functional Models (BFM).

The EDK provides a BFM simulation platform to help you simulate your peripheral. Indicate if you want this tool to generate the appropriate HDL and Bus Functional Language (BFL) stimulus file for the target bus.



☐ Generate BFM simulation platform for ModelSim-SE or ModelSim-PE

This feature requires that you have accepted the associated IBM license agreement and installed the BFM package. The link below shows how:

BFM Package Installation Instructions

No necesitamos generar los BFMs ya que no usaremos ModelSim

More Info | < Back | Next > | Cancel

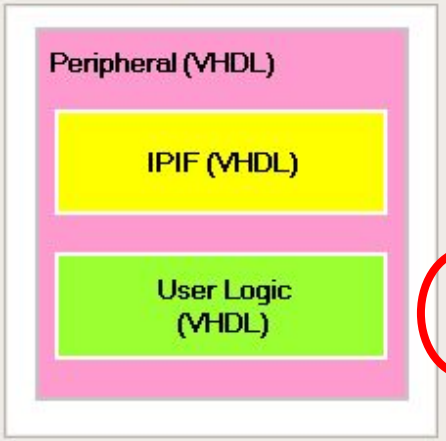
hipolito - File Browser | Xilinx Platform Studio ... | Create Peripheral

Create Peripheral

(OPTIONAL) Peripheral Implementation Support

Generate optional files for hardware/software implementation

Upon completion, this tool will create synthesizable HDL files that implement the IPIF services you requested. A stub 'user_logic' module will be created. You will need to complete the implementation of this module using standard HDL design flows. The tool will also generate EDK interface files (mpd/pao) for the synthesizable templates, so that you can hook up the generated peripheral to a processor system.



Note

Should the peripheral interface (ports/parameters) or file list change, you will need to regenerate the EDK interface files using the import functionality of this tool.

- [ ] Generate stub 'user_logic' template in Verilog instead of VHDL
- [x] Generate ISE and XST project files to help you implement the peripheral using XS
- [x] Generate template driver files to help you implement software interface

More Info | < Back | Next > | Cancel

hipolito - File Browser | Xilinx Platform Studio ... | Create Peripheral

Create Peripheral

Congratulations!

When you click Finish, HDL files representing your peripheral will be generated. You will have to implement the functionality of your peripheral in the stub 'user_logic' template file.

IMPORTANT: If you make any interface changes to the generated peripheral (including peripheral name, version, ports and parameters), or any file changes (add or remove files), you will need to regenerate the EDK interface files by using this tool in the Import mode.

```
Thank you for using Create and Import Peripheral Wizard! Please find your
peripheral hardware templates under
/home/hipolito/practica/pcores/my_plb_peripheral_v1_00_a and peripheral
software templates under
/home/hipolito/practica/drivers/my_plb_peripheral_v1_00_a respectively.

Peripheral Summary:

  top name       : my_plb_peripheral
  version        : 1.00.a
  type           : PLB (v4.6) slave
  features       : slave attachment
                   user s/w registers

Address Block Summary:

  user logic slv : C_BASEADDR + 0x00000000
                 : C_BASEADDR + 0x000000FF
```

NOTE: A *.cip settings file will be created under your peripheral's "devl" folder. It can be loaded in a future wizard session to regenerate your peripheral.

Click Finish to generate your peripheral.

More Info | < Back | Finish | Cancel

Xilinx Platform Studio ... | Create Peripheral

Contenido

- Motivación
- Creación de un periférico PLB
- Otras funcionalidades
- Añadir periférico al diseño
- Modificaciones hardware
- Modificaciones software

Otras funcionalidades

Además de registros internos, podemos seleccionar:

- FIFOs,
- Memorias,
- Interrupciones

Esto nos da más opciones para conectar el periférico, no al MicroBlaze, sino a nuestro VHDL. Desde el micro todo se accede por distintas funciones software a través del PLB, pero el VHDL verá hardware distinto.

FIFOs

Create Peripheral

FIFO Service

Configure the read/write FIFO in the IPIF.

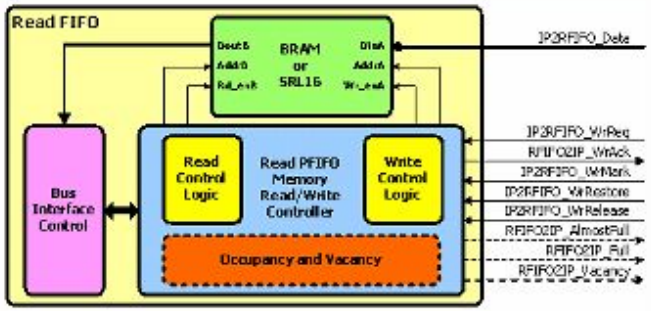
Read/Write FIFO provides data buffering service, which may automatically utilizes SRL16 primitives for the memory medium instead of BRAM primitives if packet mode is turned off and only 4 to 16 words of FIFO memory depth is needed.

☑ Include Read FIFO

☑ Use packet mode

☑ Use vacancy calculation

Number of Read FIFO entries: 512



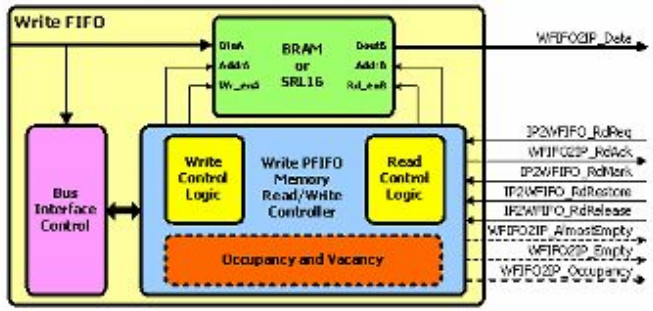
Datasheet

☑ Include Write FIFO

☑ Use packet mode

☑ Use vacancy calculation

Number of Write FIFO entries: 512



Datasheet

More Info

< Back | Next > | Cancel

Xilinx Platform Studio ... | Create Peripheral

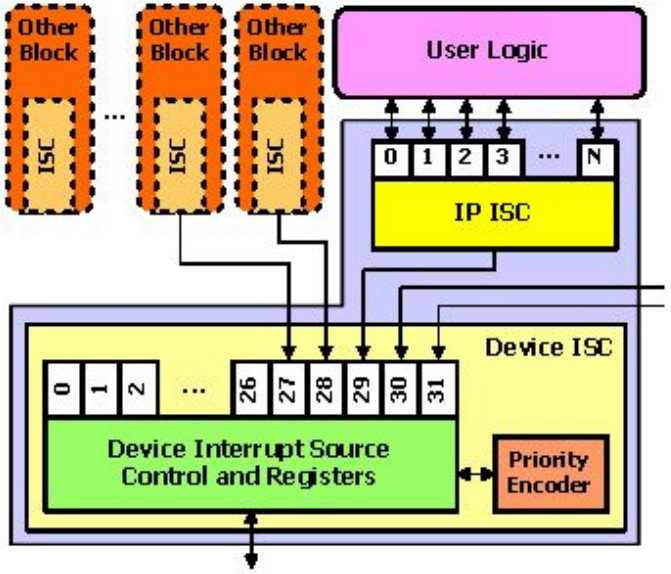
Interrupciones

Create Peripheral

Interrupt Service

Configure interrupt handling.

The interrupt control service provides interrupt capture support which captures and coalesces various interrupts generated from IPIF, other design blocks and user logic into a single interrupt output.



Datasheet

Device ISC

Device ISC (Interrupt Source Controller) coalesces all captured internal interrupts into a single output signal. You may eliminate Device ISC if all interrupts come from the user logic.

☑ Use Device ISC (interrupt source controller)

Priority Encoder

Device ISC Priority Encoder (Interrupt ID register) indicates which interrupt source has a pending interrupt. It is useful in aiding the user interrupt service routine to resolve the source of an interrupt.

☑ Use Device ISC Priority Encoder service

User logic interrrupt

Number of interrupts generated by user-logic: 1

Capture mode: Level Pass Through (non-inverted)

The input interrupt from the user logic has no additional capture processing applied to it. It is immediately sent to the IP ISC Interrupt Enable gating logic.

More Info | < Back | Next > | Cancel

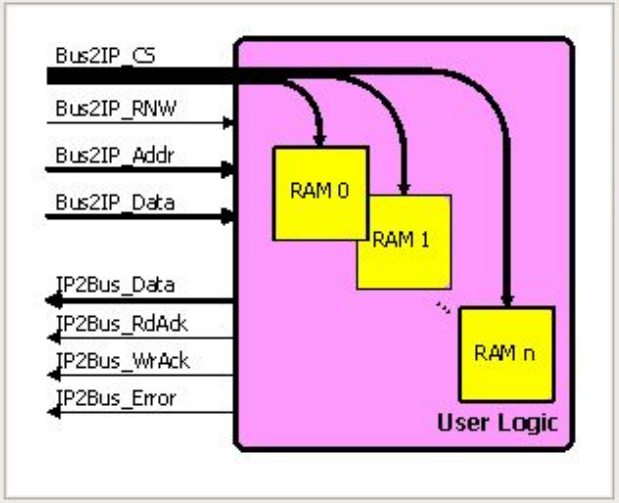
Xilinx Platform Studio ... | Create Peripheral

Create Peripheral

User Memory Space

Indicate the additional memory spaces required by your peripheral.

The user specific memory regions will be implemented in the user-logic module of your peripheral. Such memory spaces are typically seen in memory peripherals like external memory controllers. An example logic inferring multiple Block RAMs for memory read/write will be included in the user-logic module generated by the wizard tool for your reference.



User logic with memory models requires an address range CS decode, a read/write signal and the address bits to access individual memory location. The diagram on the left shows the simplest set of IPIC slave signals to access the memories.

Number of user address ranges: 1

Espacios de memoria direccionables

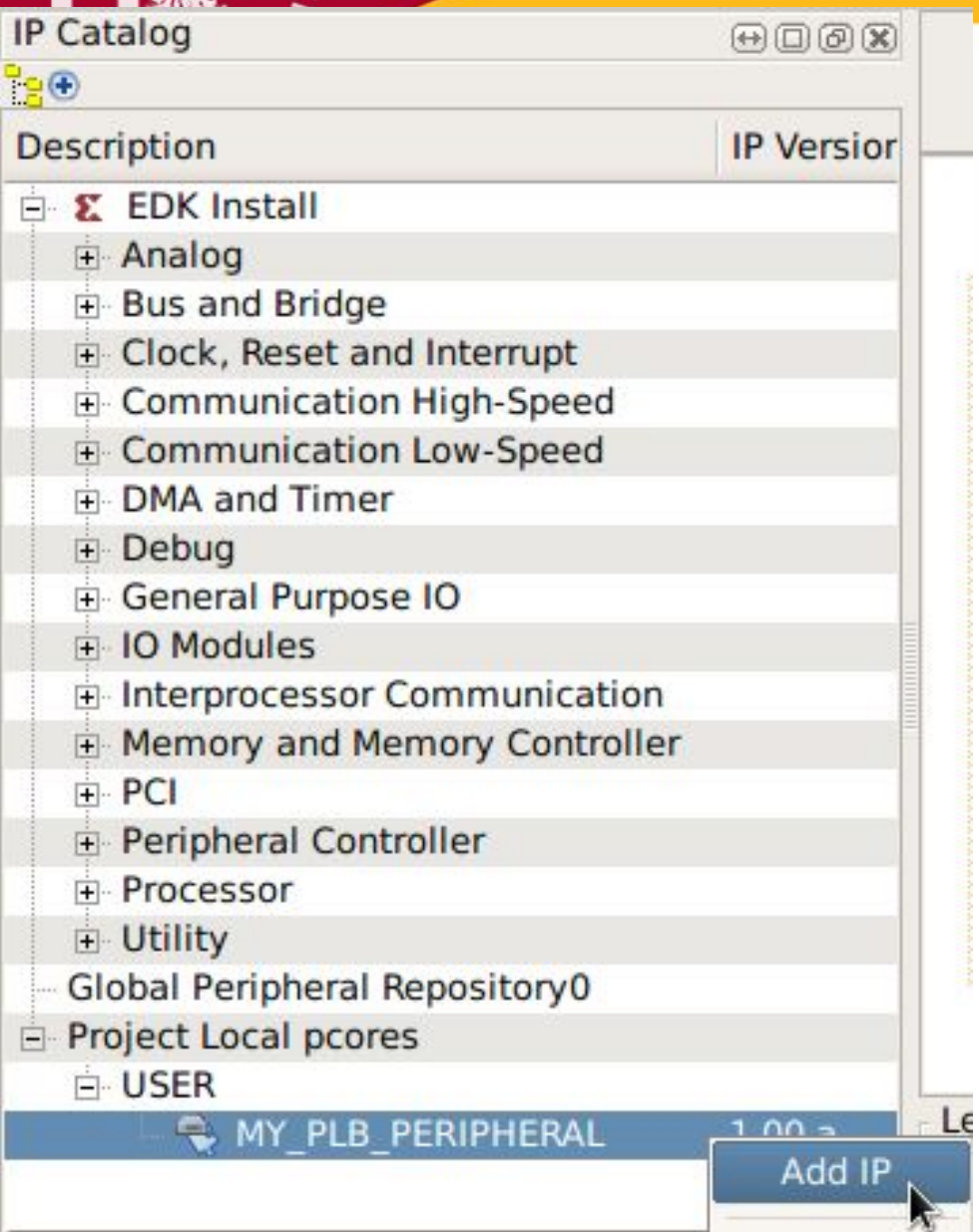
More Info | < Back | Next > | Cancel

Xilinx Platform Studio ... | Create Peripheral

Contenido

- Motivación
- Creación de un periférico PLB
- Otras funcionalidades
- Añadir periférico al diseño
- Modificaciones hardware
- Modificaciones software

Creación de un periférico PLB

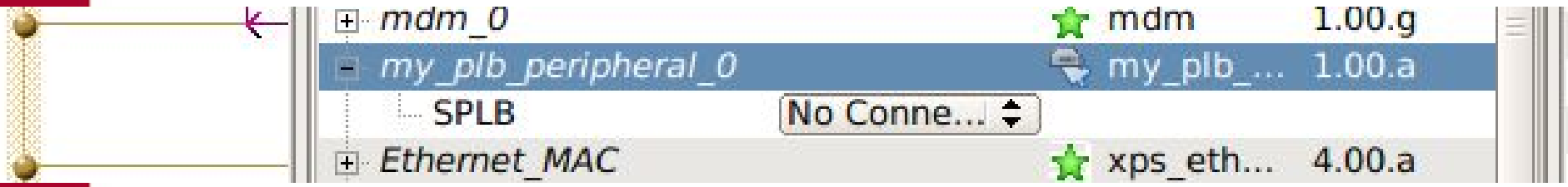


Añadir periférico al diseño

El asistente lo crea en el IP Catalog, tendremos que añadirlo como cuando añadimos el GPIO

Conexión al PLB y mapeo de direcciones de memoria

Tendremos que conectarlo al PLB del procesador y asignarle un espacio en el mapa de memoria

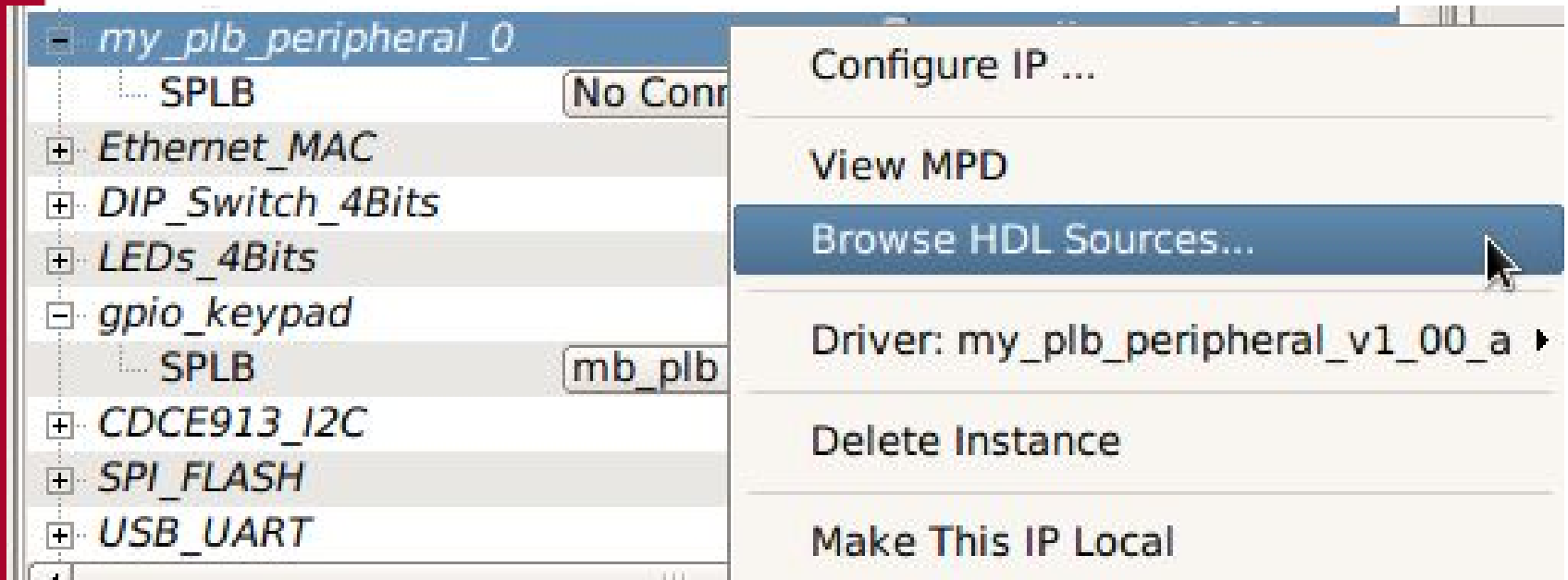


Contenido

- Motivación
- Creación de un periférico PLB
- Otras funcionalidades
- Añadir periférico al diseño
- Modificaciones hardware
- Modificaciones software

Modificaciones hardware

En el fichero `user_logic.vhd` aparece la implementación de los registros de usuario (y opcionalmente el resto de funcionalidades seleccionadas)



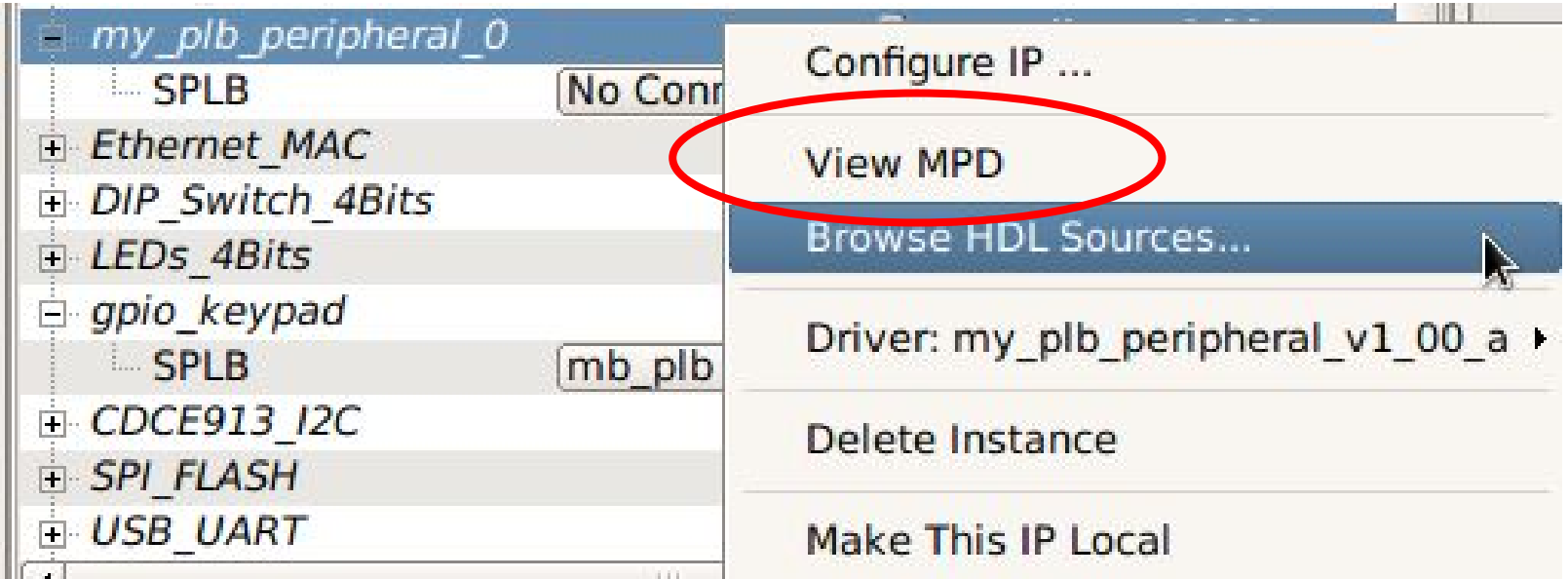
Modificaciones hardware

`user_logic.vhd`

- Leed bien
- Entended bien
- Modificad para añadir la funcionalidad del periférico

Modificaciones hardware

Si añadís pines al periférico que salen de la FPGA, tenéis que modificar el fichero MPD (Microprocessor Peripheral Definition)



Contenido

- Motivación
- Creación de un periférico PLB
- Otras funcionalidades
- Añadir periférico al diseño
- Modificaciones hardware
- Modificaciones software

Modificaciones software

El driver del dispositivo se genera en la carpeta
`<carpeta_del_proyecto>/drivers`

(si hemos marcado la opción “Generate template driver files…”)